
Trabalho de Conclusão de Curso

# Aplicação de uma Rede LoRaWAN na solução de problemas de leitura de consumo energético residencial

Alex da Silva Batista
asb@ic.ufal.br

Orientador:

Prof. Ms. Rodrigo José Sarmento Peixoto

Maceió, Janeiro de 2018

Alex da Silva Batista

# Aplicação de uma Rede LoRaWAN na solução de problemas de leitura de consumo energético residencial

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação do Instituto de Computação da Universidade Federal de Alagoas.

Orientador:

Prof. Ms. Rodrigo José Sarmento Peixoto

Maceió, Janeiro de 2018

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação do Instituto de Computação da Universidade Federal de Alagoas, aprovada pela comissão examinadora que abaixo assina.

---

Prof. Ms. Rodrigo José Sarmento Peixoto - Orientador
Instituto de Computação
Universidade Federal de Alagoas

---

Prof. Dr. Rafael de Amorim Silva - Examinador
Instituto de Computação
Universidade Federal de Alagoas

---

Prof. Dr. Leandro de Melo Sales - Examinador
Instituto de Computação
Universidade Federal de Alagoas

Maceió, Janeiro de 2018

# Agradecimentos

Ao meu Deus e Senhor Jesus Cristo, sem o qual não conseguiria chegar até aqui, pois nele encontrei força, fé e esperança de um bom futuro. Não tenho palavras suficientes para agradecê-lo.

A minha família, que esteve comigo durante todos estes anos, principalmente aos meus pais João Batista e Marlene Batista que me ensinaram grandes valores e nas suas limitações lutaram pelo melhor para seus filhos. São exemplos para mim.

Aos amigos que fiz durante a graduação, agradeço por todos os momentos que passamos juntos, foram horas de estudo e experiências, a eles meus votos de sucesso.

Aos professores do Instituto de Computação, em especial aos professores de Engenharia de Computação que me acompanharam durante toda essa caminhada. Meu agradecimento especial ao professor Rodrigo José Sarmento Peixoto que nos acompanhou desde o primeiro período até hoje e nos ensinou muito além dos conteúdos das disciplinas e ao Prof. Thiago D. Cordeiro pelo estímulo que me foi passado para permanecer no curso durante todas as dificuldades que surgiram durante seus primeiros anos de criação.

*“Cada sonho que você deixa para trás, é um pedaço do seu futuro que deixa de existir.”*

– Steve Jobs

# Resumo

Internet das coisas (*IoT - Internet of Things*) tem sido um tema em evidência e devido ao avanço tecnológico e computacional está alcançando viabilidade suficiente para sua expansão. Há previsões sinalizando que existirão bilhões de dispositivos IoT espalhados pelo mundo nos próximos anos. Junto à Internet das Coisas surge o conceito de Cidades Inteligentes (*Smart Cities*) que de modo geral é uma ampliação do IoT sendo aplicado no âmbito de cidades inteiras.

Neste contexto, onde os componentes da rede estão distantes entre si, é crucial que os links de comunicação possam transferir informações a longas distâncias com baixo consumo de energia. A utilização de tecnologias conhecidas como *Low Power Wide Area* (LPWA) têm se mostrado uma excelente alternativa para *Smart Cities*, pois fornecem o aparato necessário para a difusão dos dados na rede. Dentre as tecnologias LPWA atualmente disponíveis, a modulação LoRa em conjunto com o protocolo de comunicação LoRaWAN vêm ganhando destaque nos últimos anos. Por ser *Open source*, baseado em uma modulação robusta e a construção dos *end-nodes* serem relativamente baratos, o protocolo LoRaWAN tem despertado o interesse de diversas comunidades de desenvolvedores.

Neste trabalho, é apresentada uma visão geral acerca da rede LoRaWAN, seus componentes, segurança na rede, tipos de mensagens que trafegam na rede, chaves e identificadores utilizados para ativação de um nó. As informações acerca protocolo LoRaWAN apresentadas neste trabalho são referentes a versão 1.0.2 da especificação. Por fim, é apresentado um estudo de caso que exemplifica os potenciais benefícios que a cobertura de uma rede LoRaWAN pode trazer para uma cidade. Neste estudo de caso, a solução proposta é constituída de um *end-node* medidor de energia residencial que periodicamente envia informações acerca do consumo de energia, Servidor de Rede e Servidor de Aplicação, essenciais para o funcionamento da rede LoRaWAN, e uma aplicação web onde é possível visualizar o consumo mensal acumulado.

**Palavras-chave**:LoRa, LoRaWAN, Internet das coisas, Cidades Inteligentes

# Abstract

Internet of Things has been a topic in evidence and due to the technological and computing progress is reaching enough viability for its expansion. There are predictions signaling that there will be billions of IoT devices spread around the world in the next few years. Together with Internet of Things emerge the concept of Smart Cities, which in a general way is an extension of the IoT being applied to entire cities.

In this context, where the components of the network are distant from each other, it is crucial that the communication links can transfer information at long distances with low power consume. The use of technologies known as LPWA (Low Power Wide Area) has shown himself as an excellent approach to Smart Cities because it provides the necessary apparatus to spread the data in the network. Among the LPWA technologies currently available, the LoRa modulation together with the LoRaWAN communication protocol has been gaining prominence in the last few years. By the fact that is opensource, based on a robust modulation and the node construction be relatively cheap, the LoRaWAN protocol has aroused the interest of various communities of developers.

In this work is presented an overview of the LoRaWAN network, their components, network security, types of messages that travel in the network, keys and identifiers used in the activation of an end-node. The information about the LoRaWAN protocol presented in this work is referent to the 1.0.2 version of the specification. At the end, it is presented a case study that exemplifies the potential benefits that the coverage of a LoRaWAN network can bring to a city. In this case study, the proposed solution is composed of a residential energy meter node that periodically sends information about the energy consumption, Application Server and Network Server, essentials to the network operation, and a web application where is possible to view the accumulated monthly consumption.

**Keywords**: LoRa, LoRaWAN, Internet of Things, Smart Cities

# Conteúdo

# Lista de Figuras

# Lista de Tabelas

# 1

# Internet das coisas

## 1.1 Introdução

Internet das Coisas (*Internet of Things - IoT*) é uma área da computação que vem ganhando bastante destaque nos últimos anos e diversos fatores vêm contribuindo para que isso aconteça. Temos presenciado uma grande evolução na produção do hardware. Ao longo dos anos, foi possível observar avanços na sua miniaturização e aumento do seu potencial de processamento. Não obstante, os custos de tais dispositivos estão cada vez menores, possibilitando que um maior número de pessoas possam construir suas próprias soluções e desbravar o universo *IoT*. Estima-se que até 2020 teremos cerca de 50 bilhões de dispositivos *IoT* espalhados pelo mundo (Augustin et al., 2016).

A Internet das coisas difere da Internet convencional em diversos aspectos. Na Internet convencional, grande parte dos nós da rede são dispositivos com grande poder de processamento, poucas limitações acerca do consumo de energia, podem desempenhar diversas funções e realizar transferência de dados com grandes velocidades. Na Internet das Coisas, no entanto, os recursos dos dispositivos, tais como: tamanho físico, memória, capacidade de processamento, bateria, taxa de transferência de dados, custo, etc. são bastante limitados. Em geral, os dispositivos numa rede *IoT* são embarcados e comumente passam desapercebidos pelos seus usuários.

Atualmente, não existe nenhum padrão de arquitetura bem estabelecido para *IoT*, porém uma solução típica para a construção de uma rede, em geral possui os seguintes elementos (Eclipse.org, 2016):

- *End-nodes*, também conhecidos como *end-devices*, *nodes* ou nós.
- *Gateway*, também conhecido como *concentrator*, concentrador ou *base station*.
- *IoT Cloud Platform*

- Aplicações

Um exemplo de estrutura de uma rede IoT, é apresentada na Figura 1.1.

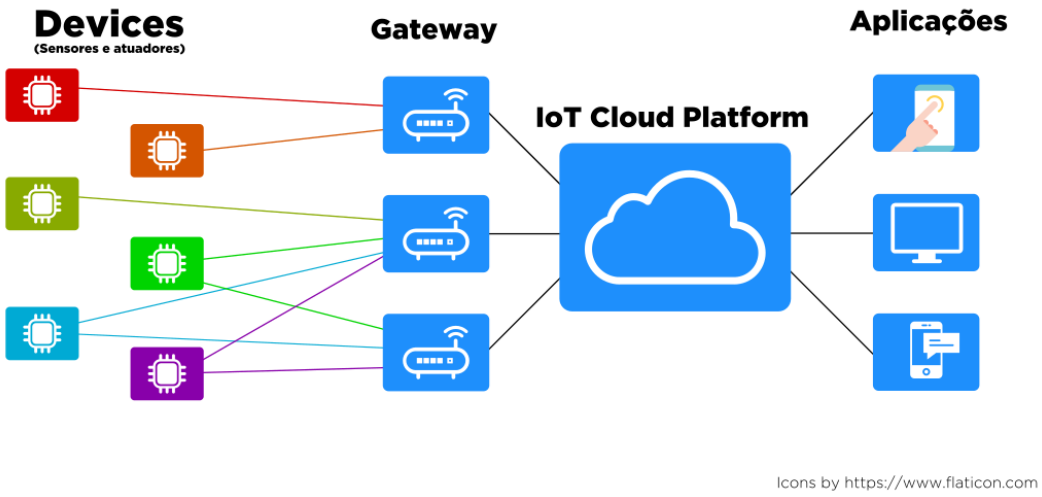


Figura 1.1: Arquitetura de uma Rede IoT

## 1.2 Tecnologias *Low Power Wide Area*(LPWA)

Grande parte da rede numa solução *IoT* é composta de comunicação sem fio, visto que frequentemente os *end-nodes* são dispositivos embarcados onde o uso de cabos ou fios para realizar as comunicações é inviável. Portanto, o desenvolvimento de novas tecnologias sem fio e a evolução das já existentes têm grande relevância na evolução do *IoT*.

As tecnologias mais populares de comunicação sem fio, embora já bem difundidas, não são apropriadas para todos os tipos de situações. Utilizar tecnologias como WiFi,*Bluetooth*, 3G ou 4G, muitas vezes não se justifica, seja por custos financeiros, limitações na área de cobertura, ou ainda pelo consumo de energia que tais tecnologias dispendem para realizar a comunicação.

O advento da internet das coisas trouxe consigo a demanda por tecnologias de comunicação *wireless* de baixo consumo de energia com sinal de alcance superior às popularmente conhecidas.Tais tecnologias são conhecidas como *Low Power Wide Area* (LPWA). Tecnologias LPWA não são, de fato, novidade, porém com o avanço do *IoT* essa modalidade vem ganhando mais atenção nos últimos anos, o que contribui para o seu desenvolvimento e popularidade. Na Figura 1.2 são apresentadas algumas tecnologias sem fio atualmente disponíveis e suas principais diferenças em termos de consumo de energia e área de cobertura. Conforme ilustrado, a tecnologia LPWA atende casos onde é necessário realizar comunicações a longas distâncias com baixo consumo de energia. Em contrapartida, os links de

comunicação na rede, limitam-se ao tráfego de pequenas quantidades de dados algumas vezes durante o dia a uma taxa de transferência relativamente baixa. Considerando que tais dispositivos são alimentados por bateria, esta deve ter duração de alguns anos.

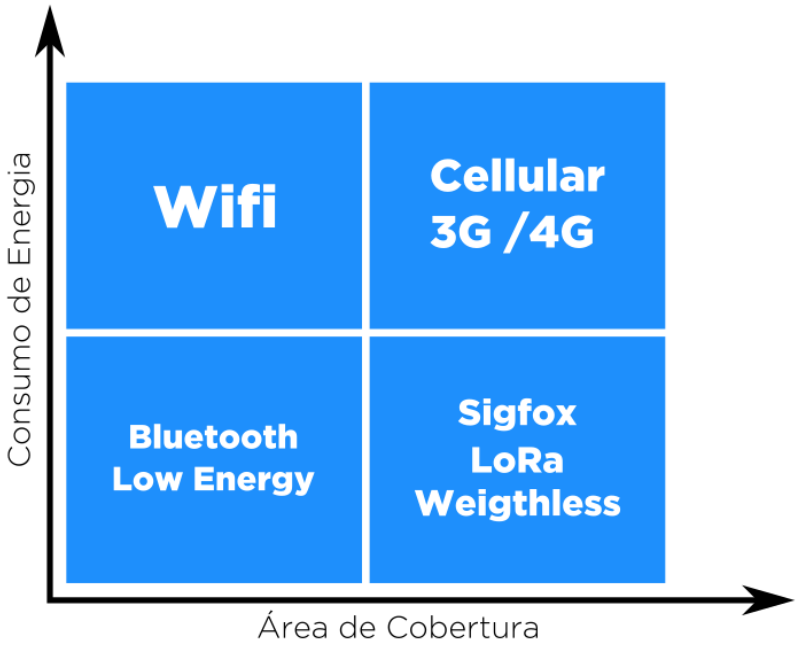


Figura 1.2: Relação entre consumo de energia e área de cobertura das tecnologias *wireless* atuais

Dentre as diversas tecnologias LPWA que existem atualmente, algumas vêm se destacando e expandindo mercado, cada uma delas com o objetivo de se tornar o padrão de comunicação para essa categoria de aplicação. Dentre tais tecnologias podemos citar: Sigfox, *Weigthless*, Ingenu e LoRa (Labs):

- Sigfox (Sigfox): Sigfox é uma tecnologia proprietária desenvolvida pela empresa francesa Sigfox que utiliza modulação *Ultra Narrow Band* (UNB) como base para sua comunicação. Seu modelo de negócios prevê apenas uma operadora responsável pela distribuição e expansão da rede Sigfox por país, conhecida como *Sigfox network operator*. A rede Sigfox possui topologia do tipo estrela e funciona na faixa ISM (*Industrial, Scientific, and Medical radio band*), cada mensagem tem largura de 100Hz com taxa de dados de 100 ou 600 bits por segundo, dependendo da região. A quantidade máxima de informação a ser transferida é de 12 bytes para mensagens enviadas pelo *end-node* para a rede, e de 8 bytes para mensagens enviadas da rede para um *end-node*.

- *Weightless* (SIG): *Weightless* é tanto o nome da tecnologia quanto do grupo responsável por ela, *Weightless Special Interest Group* (SIG). Opera na faixa de frequência de

sub-1GHz tanto licenciada como não-licenciada (ISM/SRD). *Weightless* usa canais de 12.5kHz e oferece *datarates* de 200bps a 100kbps atingindo cerca de 2km em área urbana. Oferece 3 padrões abertos: *Weightless-N, Weightless-P* e *Weightless-W*, onde diferem no consumo de energia (tempo de vida da bateria) do dispositivo e custos da rede.

- *Ingenu* (Ingenu, a): Utiliza uma tecnologia *wireless* proprietária chamada RPMA *Random Phase Multiple Access*. Opera na faixa de 2.4GHz e promete baixo consumo de energia. Como exemplo de uso, estima-se que um end-node enviado 50kb de dados a cada duas horas, alimentado com bateria de 3.3Ah, esta duraria cerca de 14.2 anos (Ingenu, b).

Além das tecnologias LPWA anteriormente citadas, existem diversas outras disponíveis, como por exemplo: *Symphony Link*, *Nwave*, 4G-LTE, *NB-IoT*, *6LoWPAN*, LTE-M1, *M-Bus*.

Neste trabalho, a modulação LoRa é apresentada em conjunto com o protocolo de comunicação LoRaWAN versão 1.0.2. Este protocolo foi escolhido por ser *open source*, baseado em uma modulação robusta, os custos inerentes a construção de uma rede LoRaWAN privada são relativamente baixos e possuir uma comunidade bastante ativa, o que facilita seu desenvolvimento e popularidade.

O principal objetivo deste trabalho é apresentar os benefícios e potenciais avanços tecnológicos que uma cidade pode usufruir com a cobertura de uma rede LoRaWAN. Tal cobertura pode viabilizar projetos de cidades inteligentes e conectadas, a um custo relativamente baixo. Como forma de exemplificar o potencial dessa abordagem, é apresentado um caso de uso de uma rede LoRaWAN onde os *end-nodes* são medidores de energia residencial. Tais medidores podem contribuir para a diminuição dos custos das concessionárias de energia elétrica além de eliminar erros de leitura que potencialmente são inseridos pelo homem.

# 2

# Rede LoRaWAN

## 2.1 Modulação LoRa

LoRa - acrônimo para *Long Range* - é um tipo de tecnologia de modulação utilizada para criar links de comunicação de longo alcance. Foi desenvolvida pela empresa francesa *Cycleo* em 2010 e adquirido pela *Semtech* em 2012 (Ducrot et al., 2016b), empresa que atualmente possui os direitos sobre a tecnologia. A modulação LoRa funciona na faixa de frequência livre - *Industrial, Scientific and Medical* (ISM) - e utiliza um esquema de modulação *Spread Spectrum* que é derivado da modulação CSS (*Chirp Spread Spectrum*) (Semtech, 2015b). Diversos *datarates* são implementados na modulação LoRa utilizando os chamados *Orthogonal Spreading Factor*. Mensagens enviadas através de diferentes *Spreading Factor*, ao mesmo tempo, dentro de um mesmo canal não colidem, assim, utilizando essa abordagem o LoRa cria dentro de um canal os chamados "canais virtuais", o que aumenta efetivamente a capacidade da rede.

A modulação LoRa possui as seguintes características:

1. **Largura de Banda escalável**: Através da modulação LoRa é possível trabalhar tanto em aplicações que utilizem faixas de frequências de banda estreita como em banda larga.

2. **Envelope constante/*Low Power***: Utiliza um envelope constante para a modulação além de ser altamente eficiente e possuir baixo consumo de energia.

3. **Robustez**: Uma das características mais importantes nesta modulação, é sua alta capacidade de resistência à interferência de ruídos.

4. **Resistente a *Multipath*/*fading***: Por utilizar *chirp pulse*, que é relativamente de banda larga, LoRa é imune a *multipath* e *fading*.

5. **Resistente ao efeito Doppler**: O deslocamento em frequência causado pelo efeito doppler introduz um deslocamento desprezível no tempo do sinal banda base.

6. **Longo Alcance**: Através da modulação LoRa é possível atingir longas distâncias com excelente robustez, podendo atingir 5Km em área urbana e 15Km em área rural.

7. **Capacidade da Rede Melhorada**: Como o LoRa utiliza *orthogonal spreading factors*, isso garante à rede que múltiplos sinais possam ser transferidos ao mesmo tempo no mesmo canal sem que um interfira no outro. Sinais modulados a um *spreading factor* diferente do que o esperado é visto pelo receptor como ruído e pode ser tratado como tal.

8. ***Ranging / Localization***: LoRa é ideal para aplicações que requerem serviços de localização devido a sua habilidade de discriminar linearmente frequência e tempo de erros.

A modulação LoRa é proprietária, portanto, caso se queira utilizar esta tecnologia em alguma aplicação será necessário a aquisição de um chip de modulação provido pela *semtech*, podendo ser encontrado no mercado em vários modelos, com diferentes configurações e faixas de frequência de operação, alguns deles são:

- *SX1272 Low Power RF Transceiver 860 - 1000MHz*
- *SX1276 Low Power Long Range Transceiver 137 MHz to 1020 MHz*
- *SX1301 Base Band Processor for Data Concentrator for Long Range Communication Network* (sem)

Os *end-nodes* comumente são construídos utilizando o SX12XX, enquanto que os concentradores utilizam o SX12XX em conjunto com o SX1301.

## 2.2 Visão Geral de uma rede LoRaWAN

Embora muitas vezes os termos LoRa e LoRaWAN sejam utilizados indistintamente, eles são diferentes. LoRa se refere a um tipo de tecnologia de modulação que habilita links de comunicação em longas distâncias, enquanto que LoRaWAN define a arquitetura da rede e um protocolo de comunicação que utiliza como base a modulação LoRa. (Alliance, 2015) Para melhor compreensão, pode-se fazer um comparativo com a pilha de protocolo da Internet, onde na Figura 2.1 nota-se que a modulação LoRa está no nível da camada física, e o LoRaWAN pertence à camada de enlace.

É possível realizar comunicações ponto a ponto apenas com LoRa, no entanto deve-se levar em conta que os pacotes não serão interpretados por uma rede LoRaWAN, pois a construção de um pacote LoRa é diferente de um pacote LoRaWAN.

A rede LoRaWAN é adequada para situações onde o *end-node* precisa enviar apenas alguns bytes de dados poucas vezes durante o dia e o consumo de energia é um fator crítico

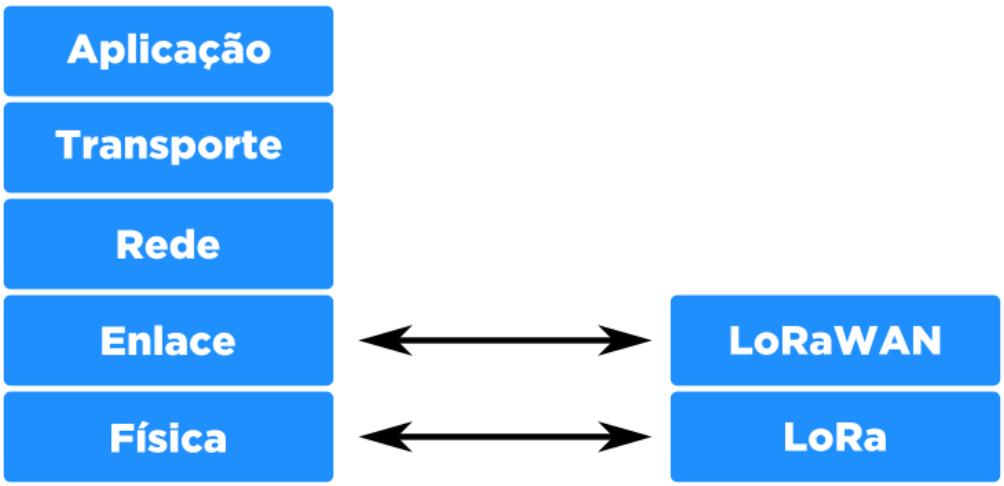


Figura 2.1: Comparativo entre a pilha de protocolo da internet (Kurose and Ross, 2013) e o LoRaWAN

para o *end-node*. A rede LoRaWAN não é adequada para situações onde o fluxo de dados é intenso ou a taxa de transferência necessária é alta.

## 2.3 Arquitetura de uma rede LoRaWAN

Uma rede LoRaWAN possui tipicamente uma topologia do tipo estrela (Alliance, 2015), onde o fluxo de comunicação ocorre apenas entre os *gateways* (também chamados de concentradores ou base *stations*) e os *end-nodes*. Não há nenhuma interação dos *end-nodes* entre si. Os diversos *gateways* apontam para o mesmo *Network Server* comunicando-se através de protocolo TCP/IP. O *Network Server*, por sua vez, comunica-se diretamente com o *Application Server*, também através de protocolo TCP/IP. As tecnologias utilizadas podem ser, por exemplo, 3G, 4G ou *Ethernet*.

Quando um pacote advindo do *gateway* chega no *Network Server*, ele é analisado a fim de saber se é válido, pertence a algum nó registrado na rede e se o *payload* não foi alterado. A verificação do *payload* é feita através de uma checagem no campo MIC (*Message Integrity Code*) do pacote. Se tudo estiver correto, então o *payload*(ainda criptografado) é extraído do pacote e transferido para o *Application Server*. A Figura 2.2 dá uma visão geral acerca da arquitetura de uma rede LoRaWAN. Nela pode-se observar as relações entre os componentes da rede além do protocolo utilizado durante as comunicações.

Um grande vantagem da rede LoRaWAN é a segurança fornecida. Além da utilização de SSL(*Secure Sockets Layer*) nas comunicações que envolvem TCP/IP, o sigilo da mensagem é

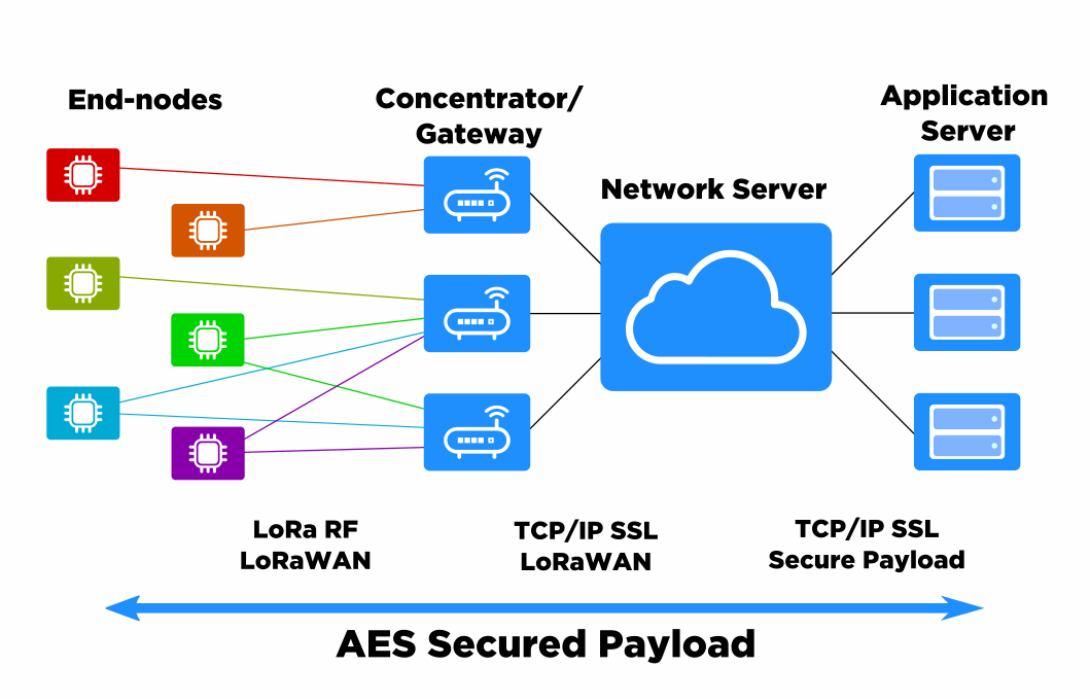


Figura 2.2: Arquitetura de uma Rede LoRaWAN

garantido ao longo de todo o seu tráfego (*end-to-end*), partindo do *end-node* até o *Application Server* (e vice-versa). A segurança da rede é garantida através de um conjunto de chaves que identificam os *end-nodes* na rede e criptografam a mensagem a ser enviada. A maneira com a qual essas chaves são atribuídas aos *end-nodes* depende do tipo de ativação utilizada: *Activation By Personalization (ABP)* ou *Over-The-Air Activation (OTAA)*.

Dentre as diversas chaves armazenadas no *end-node*, duas são bastante importantes no processo de criptografia da mensagem: *NwkSKey (Network Session Key)* e *AppSkey (Application Session Key)*. A chave *NwkSKey* fica armazenada em dois pontos da rede: No *end-node* e no *Network Server* e é única para cada *end-node*. A chave *AppSKey* também fica armazenada em dois pontos da rede: No *end-node* e no *Application Server*, sendo também, única para cada *end-node*.

Antes de enviar um pacote, o *end-node* utiliza o *AppSKey* para criptografar o *payload* usando algoritmo AES-128. Após a criptografia da mensagem, o MIC é gerado utilizando todos os campos do pacote, incluindo o *payload* já criptografado, através do algoritmo AES-128. Este processo pode ser visto na Figura 2.3. Utilizando esta abordagem, a validação e a leitura da mensagem são realizadas em pontos separados da rede, de modo que o *Network Server* é capaz apenas de checar a validade do pacote através do MIC. O *Application Server*, por sua vez, é o único capaz de descriptografar e ler o conteúdo da mensagem (*payload*), disponibilizando-a para a aplicação do usuário.

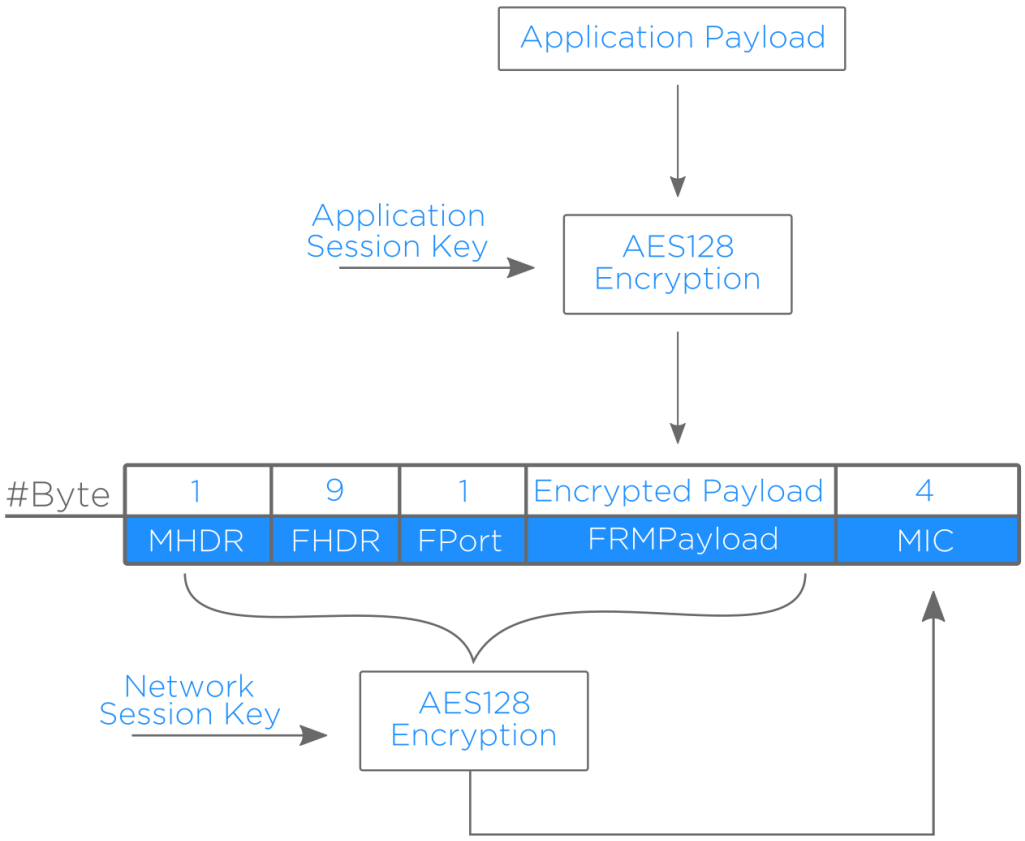


Figura 2.3: Criptografia das mensagens numa rede LoRaWAN

### 2.3.1 Mensagens de *Downlink* e *Uplink*

As mensagens na rede são classificadas em dois tipos: Mensagens de *Downlink* e mensagens de *Uplink*. *Downlinks* são mensagens enviadas pelo *Network Server* para um *end-node* específico, sendo transmitida por um único *gateway* (aquele mais próximo ao *end-node*). *Uplinks* são mensagens enviadas pelo *end-node* para o *Network Server*, sendo transmitidas através de um ou mais *gateways*. Embora a comunicação na rede seja comumente bidirecional, é esperado que mensagens de *uplink* sejam predominantes.

### 2.3.2 Mensagens Confirmadas e Não-Confirmadas

Tanto em mensagens de *Uplink* como em mensagens de *Downlink* é possível exigir do receptor uma confirmação do recebimento da mensagem (*Acknowledgment*). A esse tipo de mensagem dá-se o nome de mensagem confirmada (*confirmed message*). Quando o receptor recebe uma mensagem confirmada, ele deve responder ao emissor com um pacote que possui o bit de *acknowledgment* (ACK) setado.

Se o emissor da mensagem confirmada for um *end-node*, o *Network Server* responderá com um pacote contendo o *acknowledgment* usando uma das janelas de recepção do *end-node*. Quando o emissor da mensagem confirmada for o *gateway*, cabe ao *end-node* decidir

quando responderá com o pacote contendo o bit de *acknowledgment*.

A quantidade de retransmissões que um *end-node* fará caso não receba um *Ack*, é definida individualmente em cada um deles de acordo com seus próprios critérios. Para o *Network Server*, caso o número máximo de retransmissões seja alcançado e nenhum *Ack* do *end-node* recebido, ele considerará o *end-node* como fora da rede até que algum *uplink* deste nó seja recebido (independente do bit de *Ack* estar setado ou não).

Para mensagens não-confirmadas não há necessidade do *Acknowledgment* como resposta, porém o emissor da mensagem não terá garantia de que o receptor, de fato, ouviu sua mensagem.

### 2.3.3 *Adaptative Datarate (ADR)*

Assim que um *end-node* é autorizado a entrar na rede LoRaWAN, torna-se capaz de se comunicar de forma bidirecional dentro dela. Esta comunicação utiliza diferentes canais de frequência (dentro da faixa ISM específica para a região/país em uso) e diferentes *Datarates*. Os canais são escolhidos de forma pseudorandômica enquanto que a seleção do *Datarate* é baseada na distância entre os links de comunicação e a duração da mensagem. Um *datarate* pode ser entendido como um par relacionado entre o *spread factor* e largura de banda. Cada *datarate* possui um limite máximo de taxa de transferência. A tabela 2.1 mostra os diferentes *datarates* disponíveis na faixa de frequência US902-928MHz.

| **DataRate** | **Configuração** | **bitrate [bit/sec]** |
|---|---|---|
| 0 | LoRa: SF10 / 125 kHz | 980 |
| 1 | LoRa: SF9 / 125 kHz | 1760 |
| 2 | LoRa: SF8 / 125 kHz | 3125 |
| 3 | LoRa: SF7 / 125 kHz | 5470 |
| 4 | LoRa: SF8 / 500 kHz | 12500 |
| 5:7 | RFU | |
| 8 | LoRa: SF12 / 500 kHz | 980 |
| 9 | LoRa: SF11 / 500 kHz | 1760 |
| 10 | LoRa: SF10 / 500 kHz | 3900 |
| 11 | LoRa: SF9 / 500 kHz | 7000 |
| 12 | LoRa: SF8 / 500 kHz | 12500 |
| 13 | LoRa: SF7 / 500 kHz | 21900 |
| 14:15 | RFU | |

Tabela 2.1: US902-928 *Datarates*

O *Datarate* de um *end-node* pode ser fixo ou gerenciado pela rede, a esta última abordagem dá-se o nome de *Adaptative Datarate (ADR)*, onde a rede define de forma única para

cada *end-node*, o melhor *datarate* a ser utilizado baseado nas informações de sinal fornecidas durante a comunicação. Para que a rede controle o *datarate*, é necessário que o ADR esteja ativo no *end-node*.

## 2.4 *End-Node*

O *end-node* (também conhecido como *end-device* ou nó) é um dispositivo, geralmente embarcado, que executa tarefas específicas e é capaz de transmitir ou receber dados dentro da rede LoRaWAN. Sua construção varia de acordo com a aplicação, podendo interagir com o ambiente através de sensores e atuadores.

Um *end-node* pode possuir diferentes arquiteturas (Ducrot et al., 2016a), cada uma delas com um nível diferente de complexidade. De modo geral, a arquitetura de um dispositivo LoRa pode ser construída de acordo com a Figura 2.4.

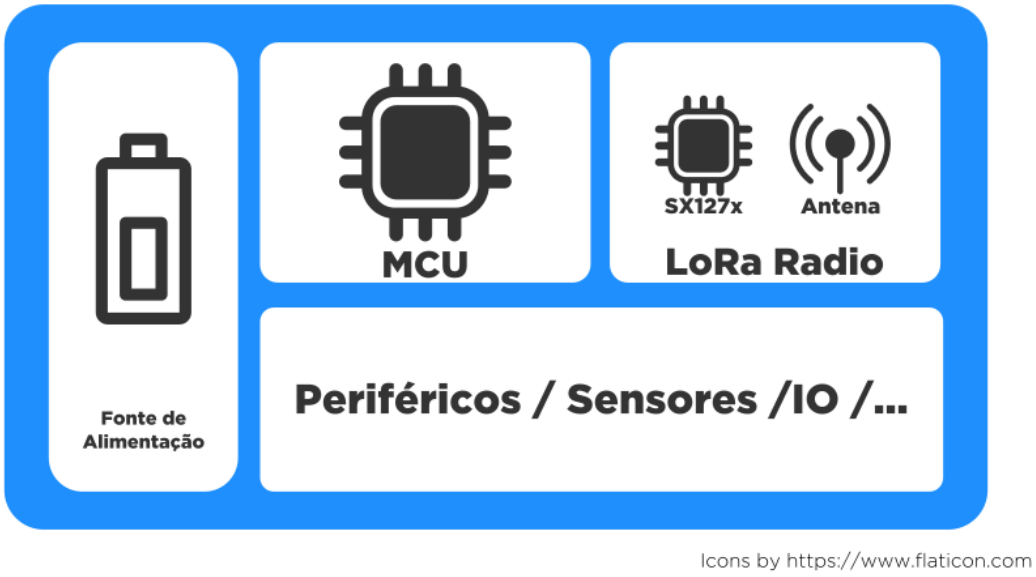


Figura 2.4: Arquitetura geral de um *end-node* LoRaWAN

O chip de modulação LoRa SX127x (representado na Figura 2.4 pelo bloco LoRa Radio) é conectado a um microcontrolador utilizando interface SPI, portanto a MCU deve obrigatoriamente dar suporte a essa interface. Caso se queira trabalhar com o protocolo LoRaWAN, além do SPI, o microcontrolador deve ter espaço de armazenamento suficiente para a pilha de protocolo, assim, deve possuir os requerimentos mínimos apresentados na Tabela 2.2 (Semtech, 2015a).

Baseado na arquitetura geral da Figura 2.4, os seguintes designs podem ser encontrados:

- Design Baseado em um *chipset* LoRa
- Design Baseado em módulo LoRa

| **Parâmetros** | **Configurações mínimas** | **Configurações recomendadas** |
|---|---|---|
| MCU | | |
| - RAM | 4KB* | 8KB |
| - FLASH | 32KB* | 64KB |
| AES 128 bits | AES decryption in software | AES Hardware decryption block |
| Radio DIOs connected to MCU IRQ inputs | DIO0, DIO1, DIO2 | DIO0, DIO1, DIO2, DIO3 |
| SPI (4 wires: SCK, MOSI, MISO, NSS) | Mandatory | Mandatory |
| RTC (32.768 kHz XTAL | Recommended for accurate time keeping | Mandatory for low-power class B implementation |
| IEEE 64-bit Extended Unique Identifier (EUI-64) | Programmed in ROM | Hardcoded in MCU |

Tabela 2.2: Requerimentos mínimos para um end-node LoRaWAN.

- Design Baseado em modem LoRa

### 2.4.1 Design baseado em um *chipset* LoRa

Este design é semelhante ao modelo geral apresentado anteriormente. A comunicação entre o microcontrolador e o *chipset* LoRa se dá através de interface SPI, além de mais 3 pinos GPIO que são utilizados para as interrupções ocasionadas pelo recebimento e envio de pacotes. Assim, são utilizados, em geral, 7 pinos para a comunicação adequada entre eles. O microcontrolador deverá conter não apenas o código da aplicação, mas também a pilha de protocolo LoRaWAN.

### 2.4.2 *Design* baseado em um módulo LoRa

Nesta abordagem, o microcontrolador e o *chipset* LoRa encontram-se encapsulados e conectados num único módulo, sendo apenas expostos os pinos que estão disponíveis para serem utilizados na aplicação. O microcontrolador deve conter o código da aplicação junto com a pilha de protocolo LoRaWAN. Utilizar um módulo LoRa como parte da solução, diminui significativamente os problemas de comunicação que poderiam ser causados por má conectividade entre o microcontrolador e o *chipset* LoRa, além do que, os *packages* dos módulos LoRa podem ser bastante pequenos, viabilizando projetos onde o tamanho da solução é um fator crítico.

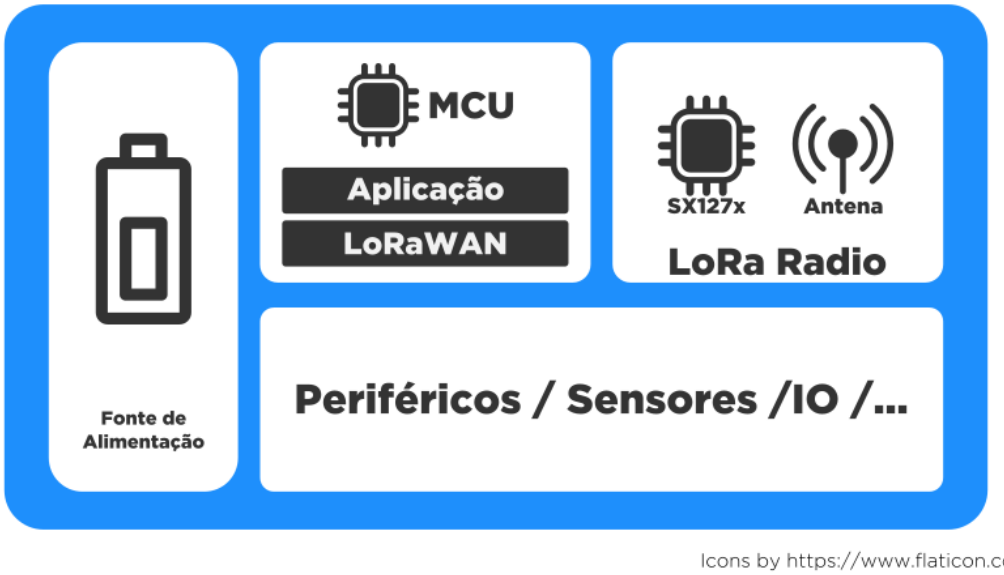


Figura 2.5: *Design* baseado em *chipset* LoRa

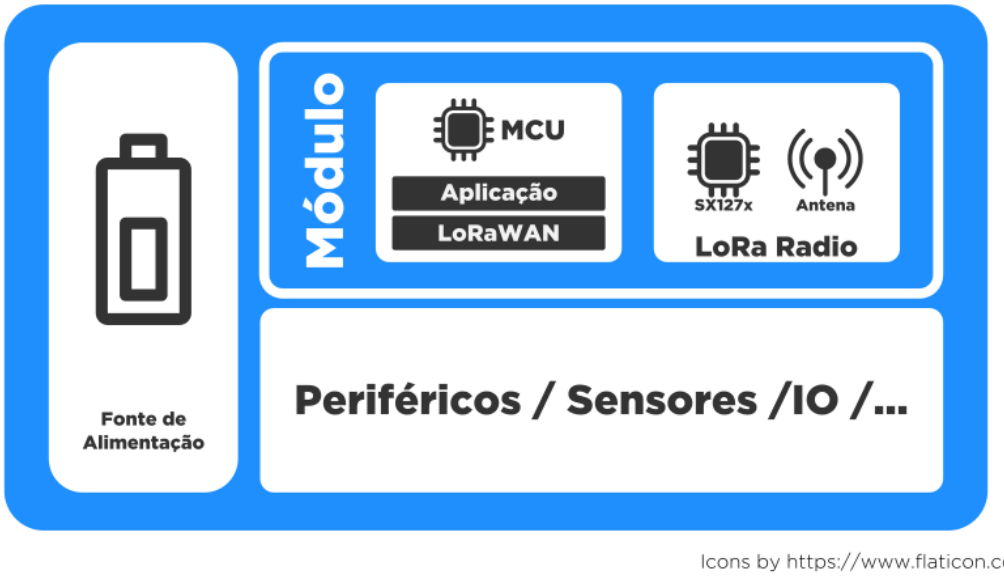


Figura 2.6: *Design* baseado em módulo LoRa

### 2.4.3 *Design* baseado em um modem LoRa

A construção de soluções utilizando um modem LoRa, pode ser considerada a abordagem mais simples a ser escolhida. Isso porque a comunicação entre o modem LoRa e o microcontrolador ocorre, normalmente, via UART (*Universal asynchronous receiver-transmitter*) através de comandos AT. O projetista ou desenvolvedor não precisa de muito conhecimento acerca do protoloco de comunicação LoRaWAN, já que o código da aplicação fica desvinculado da pilha de protocolo, sendo este implementado no modem. A desvantagem do design

modem LoRa está no custo (pois o módulo é, em geral, mais caro) e no consumo de energia.

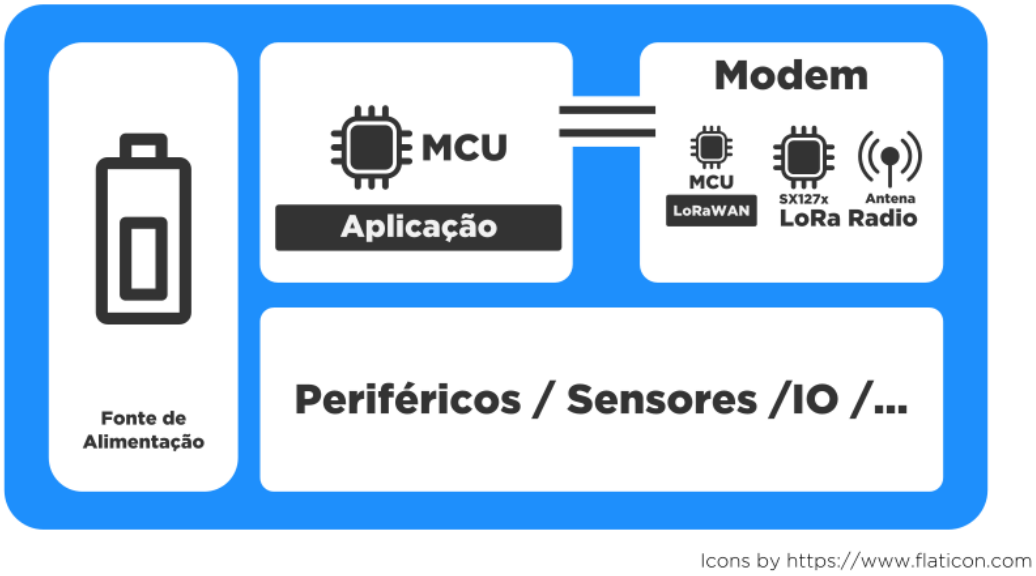


Figura 2.7: *Design* baseado em modem LoRa

### 2.4.4 Classes de um End-Node

Os *end-nodes* numa rede LoRaWAN são classificados de acordo com a forma de gerenciamento de suas janelas de transmissão e recepção. De acordo com a especificação LoRaWAN 1.0.2, os end-nodes são distribuídos em 3 Classes: A, B e C (Sornin et al., 2016).

Todos os nós, obrigatoriamente, devem implementar a classe A. As classes B e C, são incrementos baseados na classe A. Assim, é importante destacar que todo nó de classe B ou C implementa a classe A, as classes B e C não possuem relação entre si e ao nó é permitido possuir apenas uma classe por vez.

#### Classe A

*End-nodes* de classe A (Sornin et al., 2016) são os mais econômicos em termos de consumo de energia. São ideais para aplicações onde os nós são alimentados através de bateria e esta deve durar por muitos anos. A principal razão pela qual os nós de classe A são os mais econômicos, se dá pelo fato de que as janelas de recepção do radio ficam abertas apenas por um curto espaço de tempo, logo após a transmissão de algum pacote.

Os *end-nodes* na rede LoRaWAN possuem 3 janelas de comunicação: 1 TX, para envio dos pacotes, e 2 RXs para a recepção dos pacotes. A forma com a qual essas janelas de comunicação são gerenciadas varia de acordo com a classe do *end-node*, para classe A, funciona da seguinte forma: O fluxo de comunicação sempre é iniciado pelo *end-node*, enviando

um *uplink*. Após o envio do *uplink*, o *end-node* aguarda RECEIVE_DELAY1 segundos e então abre a janela de recepção RX1. Se o *end-node* não recebe nenhuma mensagem durante a abertura de RX1, a janela RX2 é aberta logo após o fechamento de RX1. Caso nenhuma mensagem seja recebida em RX2, ela também é fechada. Todas as janelas de comunicação permanecem fechadas até que o *end-node* envie novamente um *uplink*.

Caso uma mensagem seja recebida em RX1, a janela RX2 não será aberta. Após a transferência completa de uma mensagem, a janela de recepção que capturou a mensagem é então fechada. Não é possível transmitir e receber mensagens ao mesmo tempo. A duração das janelas de recepção, *RXwindow*, é uma função do *spread factor* (SF) e largura de banda (BW - *BandWidth*). Sendo definida da seguinte forma (Semtech):

$$\mathrm{RX}window = (2 \times \mathrm{Tsymb} + 2 \times \mathrm{RXerror}) \tag{2.1}$$

onde *Tsymb* é dado por

$$\mathrm{Tsymb} = 2^{\mathrm{SF}}/\mathrm{BW}segundos \tag{2.2}$$

e *RXerror* se refere ao erro máximo do tempo de abertura da janela de recepção. O receptor é ativado em um intervalo de tempo de [-RXerror: +RXerror] segundos, centrado em um *RXoffset*. O *RXoffset* diz respeito ao *offset*, em segundos, entre o tempo (ótimo) de abertura do receptor e o início da transmissão do *gateway*.

O gerenciamento das janelas de comunicação para os nós de classe A é apresentado na Figura 2.8.

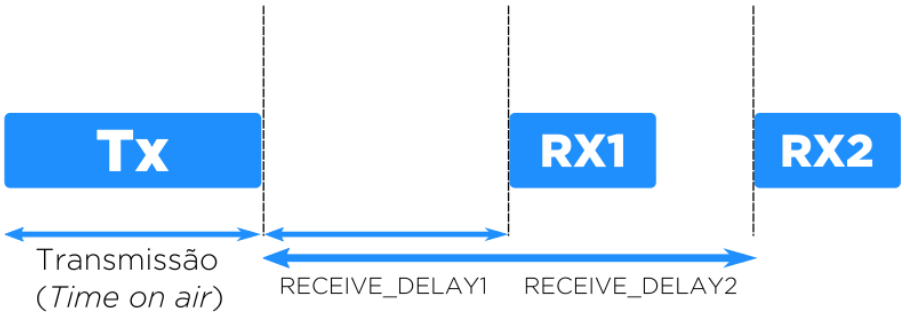


Figura 2.8: Gerenciamento das Janelas de comunicação em nós de classe A

**Classe B**

*End-devices* de classe B tentam diminuir a latência na comunicação dos *downlinks* enquanto preservam ao máximo o consumo de bateria.Um *end-node* classe B realiza a abertura da

janela de recepção não apenas após a transmissão de um pacote, como ocorre nos de classe A, mas também em tempos previsíveis. Durante a operação de um nó classe B, diversas janelas de recepção são abertas em intervalos fixos. Os *downlinks* podem ser enviados nestes intervalos.

Para que isso seja possível, o *gateway* envia *beacons* regularmente para sincronizar todos os *end-devices* (de classe B) na rede, assim o nó consegue abrir essas janelas extras de recepção (chamadas de *ping slot*) periodicamente.

Para uma rede suportar nós de classe B, todos os *gateways* precisam sincronamente enviar *beacons*, provendo assim uma referência de tempo para os *end-devices*. Baseado nisso, os *end-devices* abrem suas janelas de recepção periodicamente.

#### Classe C

Os *end-nodes* de classe C (Sornin et al., 2016) são os menos econômicos em termos de consumo de energia, no entanto possuem a vantagem de ser a classe com menor latência para comunicações de *downlinks*. São mais adequados em aplicações onde o consumo de energia não é um problema crítico.

A gerência das janelas de comunicação para *end-nodes* de classe C ocorre da seguinte forma: O fluxo de comunicação pode ser iniciado tanto pelo *end-node*, através de *uplinks*, como pela rede, através de *downlinks*. Enquanto o *end-node* estiver ligado e não estiver transmitindo nenhum dado, a janela de recepção RX2 estará aberta na maior parte do tempo. Quando um dado precisa ser transmitido, o *end-node* fecha as janelas de recepção e abre a janela de transmissão. Logo após o termino da transmissão, a janela RX2 é aberta durante RECEIVE_DELAY1 segundos, caso nenhuma mensagem seja recebida nesse intervalo, RX2 é fechada e a janela RX1 é aberta durante alguns milisegundos (Semtech), caso nenhuma mensagem seja recebida durante a abertura de RX1, RX2 abre novamente, permanecendo assim até que uma nova mensagem precise ser transmitida ou alguma mensagem seja recebida. Não é possível transmitir e receber mensagens ao mesmo tempo. O gerenciamento das janelas de comunicação para os nós de classe C é apresentado na Figura 2.9.

### 2.4.5 Identificadores e chaves de um *end-node*

Os identificadores e chaves que devem ser armazenados no *firmware* do *end-device* (antes do processo de *Join*) variam de acordo com o tipo de ativação. Se *Over-the-Air Activation* (OTAA) for utilizado, os seguintes identificadores e chaves devem estar previamente no *end-device*: *DevEUI*, *AppEUI*, *AppKey*. Após entrar na rede, *DevAddr*, *NwkSKey* e *AppSkey* são geradas para cada *end-node*. Quando se utiliza *Activation By Personalization* (ABP), todas as chaves e identificadores devem estar armazenados no *firmware end-node* (exceto *AppKey*), de modo que o procedimento de *Join* não precisa ser realizado.

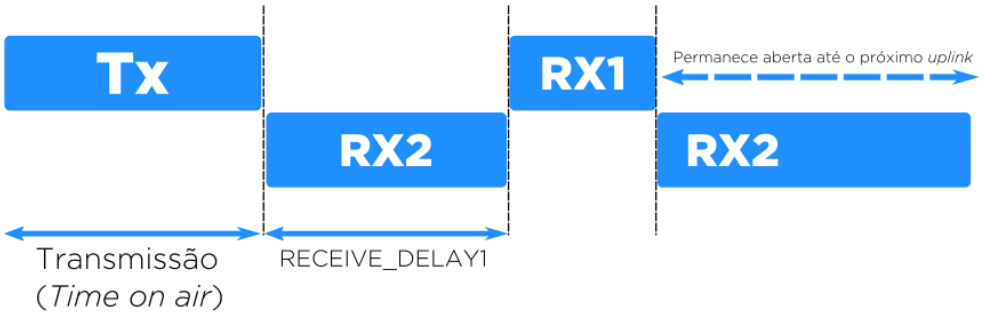


Figura 2.9: Gerenciamento das Janelas de comunicação em nós de classe C

O conjunto de identificadores e chaves que habilitam um *end-node* numa rede LoRaWAN é composto de três chaves, dois identificadores únicos e um endereço do dispositivo. Cada um deles é descrito em detalhes a seguir.

***End-device Identifier (DevEUI)***

É um ID único global no formato IEEE EUI64, que identifica o *end-device*. Este identificador não deve ser gerado randomicamente para associação a um *end-device*, pois isso não garante sua unicidade. Alguns microcontroladores já possuem este identificador único, neste caso, a aquisição do microcontrolador garante o direito de uso de seu identificador. Porém, se o microcontrolador não possuir o identificador, este pode ser adquirido posteriormente em blocos através do IEEE.

***Application Identifier (AppEUI)***

É um ID de aplicação único global no formato IEEE EUI64 que identifica a entidade capaz processar a requisição de *Join*. Este identificador deve estar presente no *end-node* antes da realização do procedimento de *Join*.

***Application Key (AppKey)***

O *Application Key* é uma chave AES-128 específica para o *end-device*. Esta chave é utilizada para gerar as chaves específicas de sessão (*Network Session Key* e *Application Session Key*) do *end-node* sempre que a ativação via OTAA for utilizada.

***End-Device Address (DevAddr)***

É um identificador (único dentro da rede) de 32bits, que o *end-node* recebe ao entrar numa rede. Possui o seguinte formato:


Figura 2.10: distribuição dos bits no *DevAddr*

Onde *NwkID* é utilizado como um identificador da rede, sua principal função é separar endereços de redes de diferentes operadoras que se sobrepõem territorialmente, evitando problemas de *roaming*. Os 25 bits restantes compõem o *Network Address*, e pode ser um valor arbitrário atribuído pela operadora da rede.

#### *Network Session Key (NwkSKey)*

É uma chave de sessão da rede, única para cada *end-device*. Deve ser atribuída manualmente no *firmware* do *end-node* quando se utiliza *Activation By Personalization*. Usando *Over-The-Air Activation*, esta chave é gerada automaticamente após o *end-node* entrar na rede. É utilizado tanto pelo *Network Server* como pelo *end-device* para calcular e verificar o MIC (*Message Integrity Code*) dos pacotes de mensagens a fim de garantir a integridade dos dados. Além disso, o *NwkSKey* é utilizado para criptografar e descriptografar o campo de *payload* de um pacote que contém apenas comandos MAC.

#### *Application Session Key (AppSKey)*

É uma chave de sessão da aplicação, única para cada *end-device*. É utilizada tanto pelo *Application Server* como pelo *end-device* para criptografar e descriptografar o campo de *payload* de um pacote que contém dados da aplicação. Deve ser atribuída manualmente no *firmware* do *end-node* quando se utiliza *Activation By Personalization*. Usando *Over-The-Air Activation*, esta chave é gerada automaticamente após o *end-node* entrar na rede.

### 2.4.6 Ativação e personalização de um *end-node*

Para participar de uma rede LoRaWAN, cada *end-node* precisa ser personalizado e ativado. A personalização diz respeito ao conjunto de identificadores e chaves que serão armazenados no *firmware* do *end-node*, enquanto que a ativação corresponde à maneira com qual o dispositivo recebe a permissão para entrar numa rede. A ativação de um *end-node* pode ser feita de duas maneiras: utilizando a abordagem *Over-The-Air Activation (OTAA)*, onde o processo de ativação do *end-node* ocorre em duas etapas ou utilizando a abordagem *Activation*

*By Personalization (ABP)*, onde as duas etapas realizadas no *OTAA* são executadas de uma só vez (Sornin et al., 2016).

#### *Over-the-Air Activation (OTAA)*

A ativação de um *end-node* via OTAA ocorre em duas etapas: Na primeira etapa o identificadores *DevEUI*, *AppEUI* e a chave *AppKey* são armazenados no *firmware* do *end-node* e registrados no *Network Server*. Tais identificadores serão utilizados na segunda etapa a fim de que o *Network Server* reconheça o *end-node* que requisita a entrada na rede e permita o acesso.

A segunda etapa, é conhecida como *join procedure* (procedimento de entrada ou procedimento de *Join*), nesta etapa, o *end-node* requisita a entrada na rede e após a verificação de seus identificadores, seu acesso à rede é permitido ou negado. Caso seja permitido, o *end-node* recebe duas chaves de sessão: *AppSKey* (*Application Session Key*) e *NwkSKey* (*Network Session Key*). Essas chaves são geradas a partir do *AppKey* e serão utilizadas na criptografia dos pacotes enviados dentro da rede. Analogamente, funciona como um *cookie* de sessão no navegador.

O *end-node* realiza o *join procedure* sempre que perder a informação do contexto da sessão. Considerando o ponto de vista de um *end-device*, o *join procedure* consiste basicamente de uma troca de comandos MAC com o *Network Server*, onde ele envia um *join request* e caso seja aceito na rede, recebe uma mensagem *join accept* proveniente do *Network Server*.

#### *Activation By Personalization (ABP)*

Quando um *end-node* é ativado por personalização (ABP), as duas etapas realizadas no OTAA são resumidas em apenas uma. Todas as chaves necessárias para que um *end-node* se comunique na rede são armazenadas diretamente no *firmware* do dispositivo, de modo que as chaves de sessão são fixas para o *end-node*. Este tipo de configuração, em geral, é mais inseguro e não recomendado, pois as chaves de sessão armazenadas no *firmware* configuram um ponto de falha na segurança do *end-node* na rede.

## 2.5 *Gateway*

O *Gateway*, também conhecido como concentrador ou *base station*, é o componente da rede que serve como interface entre a parte da rede que utiliza LoRa RF para a comunicação e a parte que utiliza TCP/IP (Ducrot et al., 2016c). É responsável pela transferência bidirecional dos pacotes entre os *end-nodes* e o *Network Server*. Deve ter seus canais configurados adequadamente, de forma que coincida com os canais configurados no *end-node*, caso contrário ocorrerá grande perda de pacotes e a comunicação se tornará inviável.

## 2.6 Servidor de Rede (*Network Server*)

O servidor de rede é o componente da rede responsável por receber os pacotes provenientes dos diversos *gateways* e realizar diversos procedimentos como:

- Filtrar pacotes duplicados. Tais duplicações acontecem quando mais de um *gateway* recebe a mensagem de um *end-node*;
- Verificar a integridade da mensagem (checagem do *payload*) através do cálculo do MIC (*Message Integrity Code*);
- Gerenciar os *gateways* que compõe a rede;
- Ativar dos *end-nodes*;
- Gerenciar o *datarate* dos *end-nodes*, quando tal gerenciamento é permitido;
- Agendar a transmissão dos *downlinks*;

## 2.7 Servidor de Aplicação *(Application Server)*

O servidor de aplicação é o componente responsável por descriptografar o *payload* de um pacote e garantir meios para que a aplicação final consuma esses dados, bem como, é responsável por criptografar um *payload* e repassá-lo para o *Network Server*. A aplicação final pode se comunicar com o servidor de aplicação, por exemplo, através de uma API REST. Numa mesma rede pode existir diversos servidores de aplicação, cada um deles lidando com um conjunto específico de *end-nodes* pertencentes a uma determinada aplicação. Por exemplo, pode-se ter um servidor de aplicação que gerencia *end-nodes* medidores de energia, outro que gerencia *end-nodes* de sensoriamento de umidade, outro gerenciando *end-nodes* de sensoriamento de temperatura, etc.

# 3

# Estudo de caso de uma rede LoRaWAN

Atualmente, a aferição de uso de bens de consumo (por exemplo, água, energia e gás) é feita de forma manual, custosa e imprecisa. A proposta apresentada neste estudo de caso, é desenvolver uma solução prática e viável para o problema de medição de consumo residencial de energia. O sistema de monitoramento de consumo proposto, quando instalado nas residências, envia leituras periódicas de consumo para a concessionária, permitindo desta forma, análise de comportamento de consumo e aferição do custo total mensal, para que sejam geradas de forma automática as contas no final de cada mês. O desenvolvimento da solução foi dividido nas seguintes etapas:

- Criação do hardware do sistema de comunicação (esquemático e placa de circuito impresso);
- Montagem das placas de circuito impresso;
- Desenvolvimento do *firmware* de comunicação a fim de enviar os dados coletados pelo medidor de energia para a nuvem;
- Configuração do Servidor de rede e do Servidor de Aplicação;
- Criação de uma aplicação web para exibição dos dados;

O hardware do sistema de medição foi produzido fora do escopo deste projeto e está apenas sendo utilizado como forma de validação da proposta. A placa utiliza um circuito integrado específico para aplicações de medição, referenciado como U1 no centro da placa na figura 3.1.

Através da placa do sistema de medição é possível determinar o consumo de tensão e corrente *rms*, potência ativa, potência aparente e potência reativa com menos 0.1% de erro na medição de energia ativa e reativa, e menos de 0.2% de erro na medição de corrente *rms* instantânea (Devices, 2017).

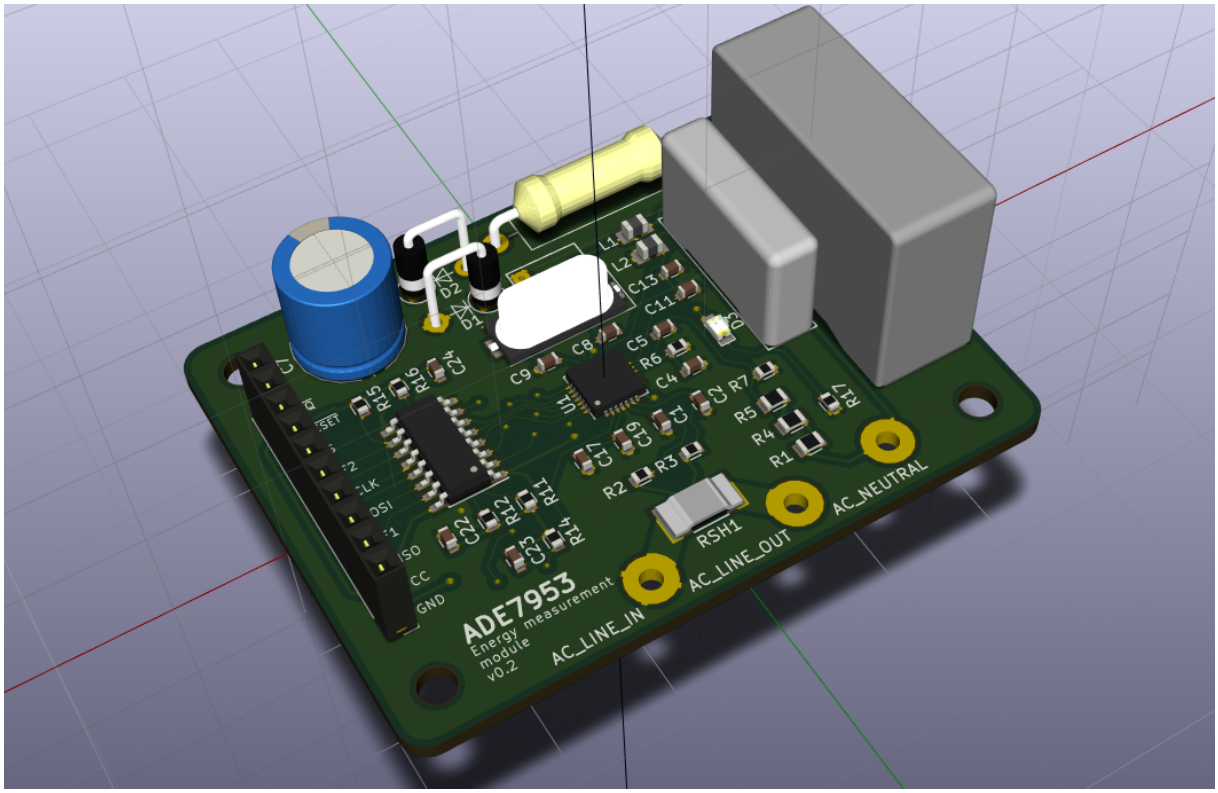


Figura 3.1: *Hardware* medidor de Energia.

## 3.1 Elaboração do Circuito de Comunicação e Placa de Circuito Impresso

A placa de comunicação utiliza um módulo LoRa provido pela *Murata Electronics* modelo CMWX1ZZABZ-078 (*Murata Type ABZ LoRa Module*). Este módulo consiste de um chip de modulação SX1276, um oscilador externo e um microcontrolador *STMicro STM32L0 series ARM Cortex M0+ 32 bit* (Murata, 2017). Para alimentação do módulo Murata, foi utilizado uma fonte com entrada 100-240VAC e saída 3.3VDC/3W provido pela *Tenstar Robot* modelo TSP-03. A placa fornece uma interface para a gravação do *firmware* e *debug* (conexão J6) e uma UART (*Universal asynchronous receiver-transmitter*) para realizar comunicação serial quando necessário (conexão J1). O esquemático do circuito de comunicação pode ser visto na Figura 3.2.

Nas Figuras 3.3 e 3.4, são exibidos respectivamente uma visão 3D da parte frontal e visão 3D da parte traseira da placa.

O *firmware* foi desenvolvido utilizando a ferramenta *CubeMX* em conjunto com a IDE *System Workbench for STM32*(SW4STM32) provido pela *STMicroelectronics*.

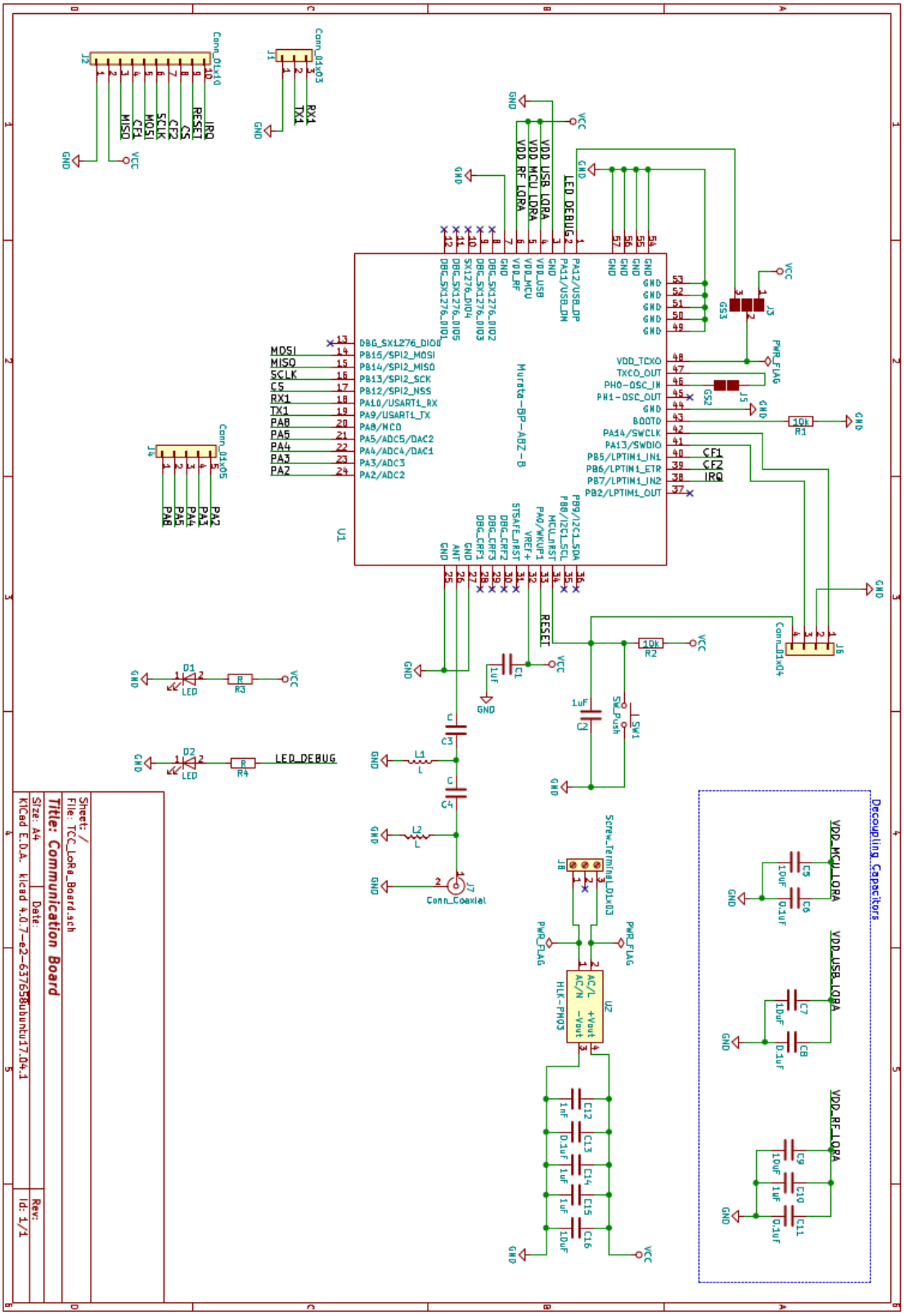


Figura 3.2: Esquemático da placa de comunicação *LoRaWAN*.

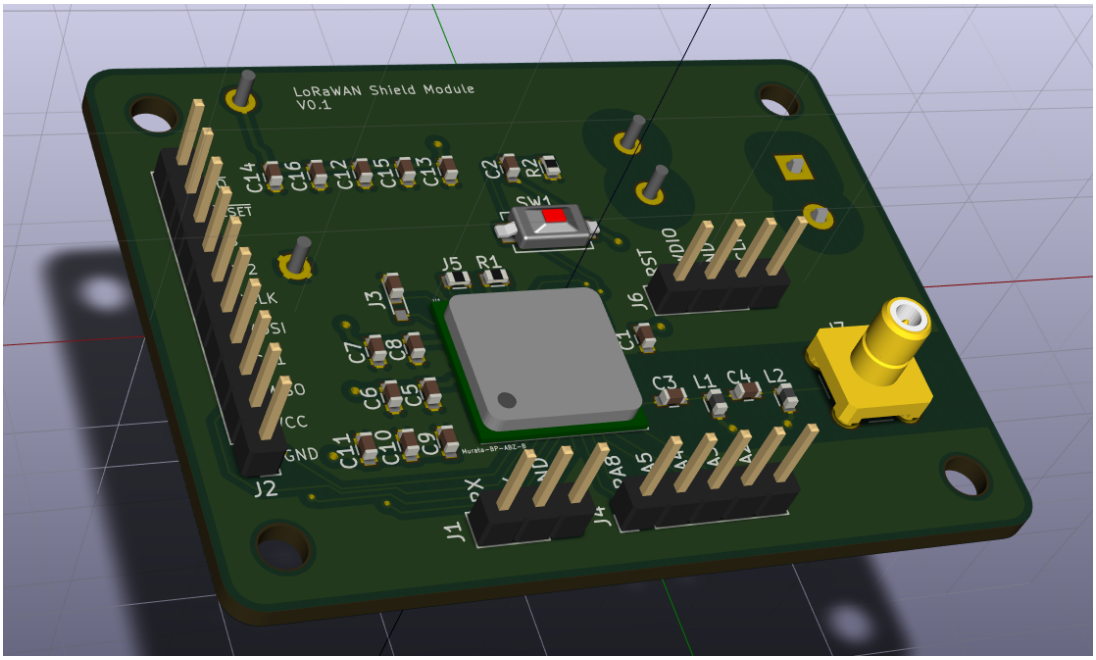


Figura 3.3: Visão 3D frontal da *Shield LoRaWAN*

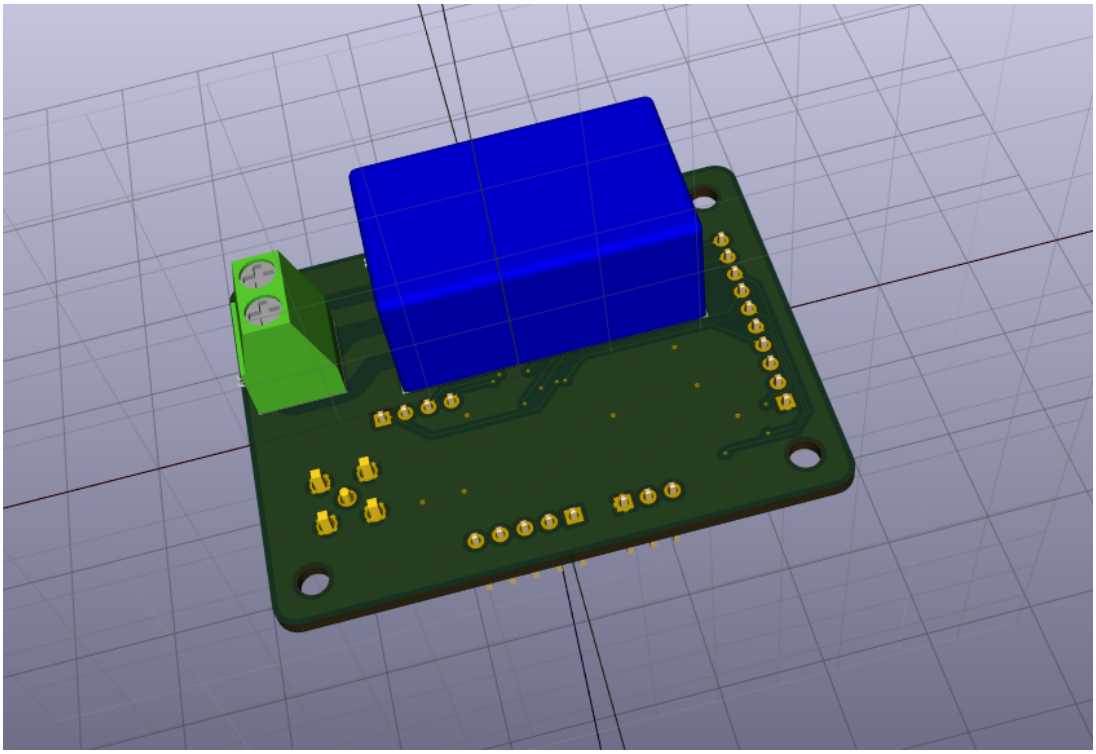

Figura 3.4: Visão 3D traseira da *Shield LoRaWAN*.

## 3.2 Configuração do Servidor de Rede e Servidor de Aplicação

Para a implementação dos servidores de rede e aplicação foi utilizado o projeto *LoRa Server* (https://www.loraserver.io/). *LoRa Server* é um projeto *open-source* disponível no github e um dos mais populares. Para sua implantação, foi criado um servidor local executando uma *docker* com ambos os servidores, de rede e de aplicação. Nas Figuras 3.5, 3.6 e 3.7 estão apresentadas as telas de gestão das aplicações e dos *end-nodes*.

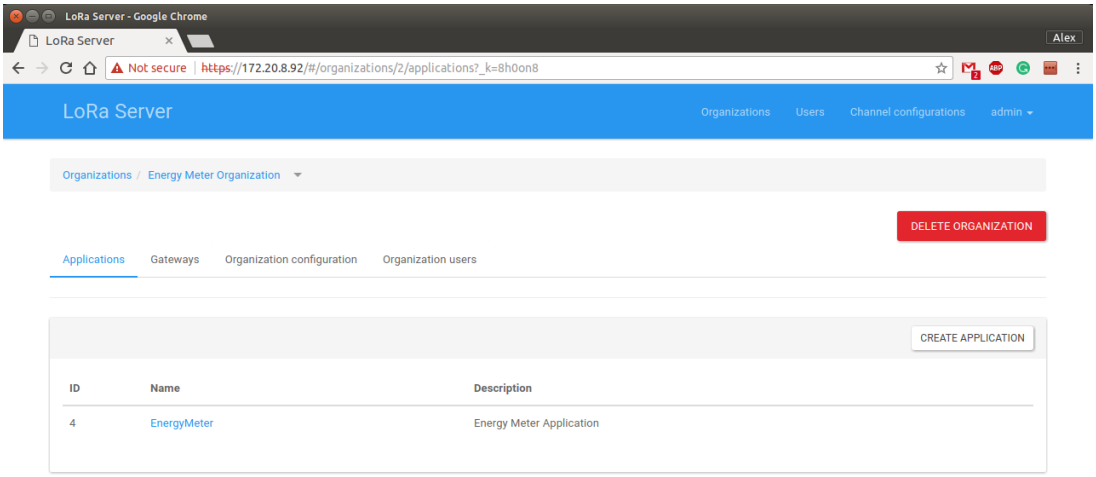


Figura 3.5: LoRa Server - Gestão das Aplicações.

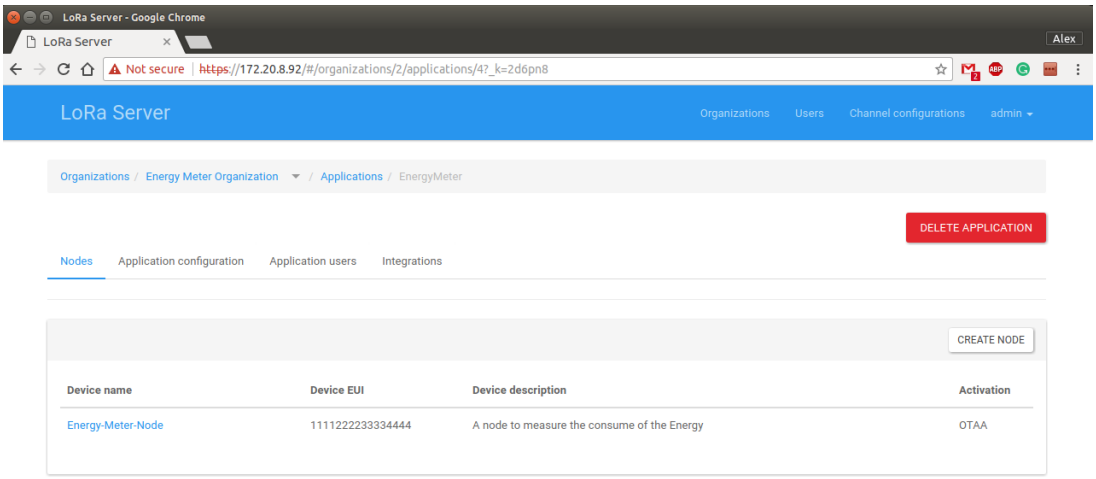


Figura 3.6: LoRa Server - Gestão dos *end-nodes*.

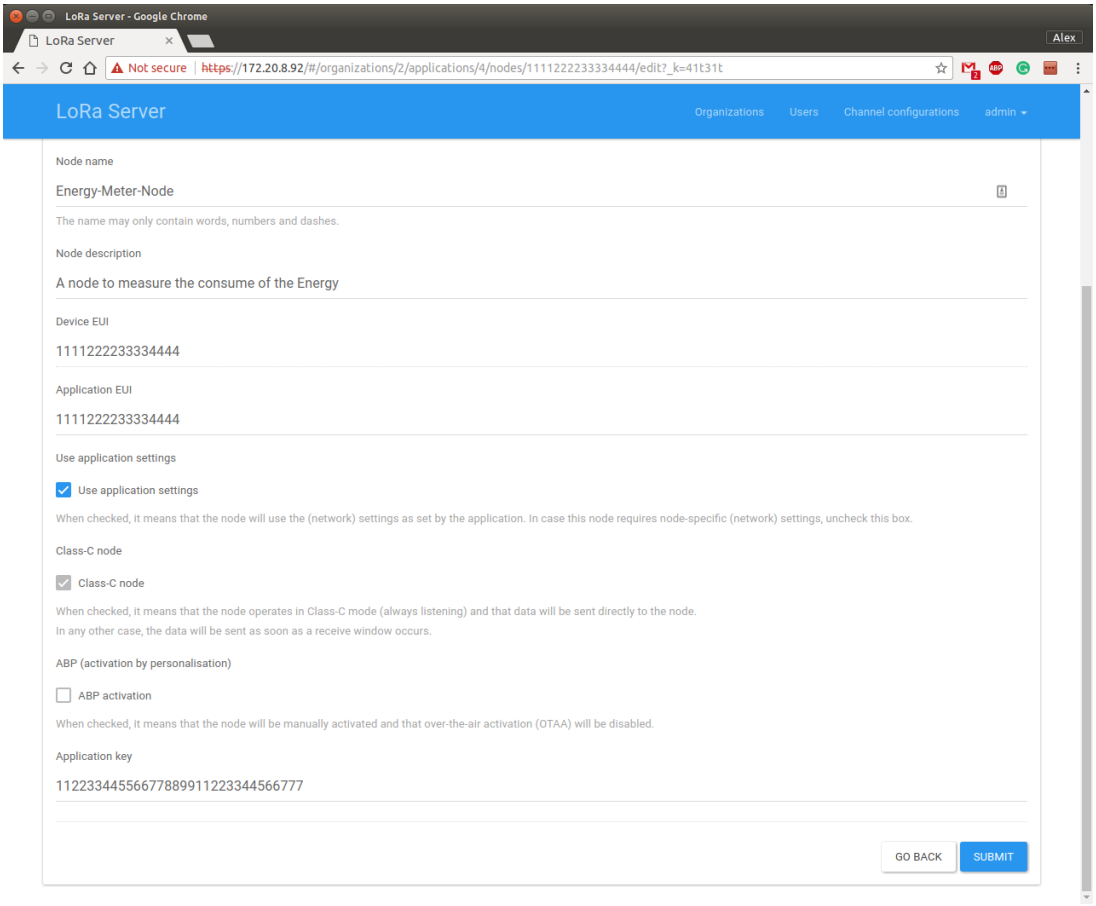


Figura 3.7: LoRa Server - Configuração de um *end-node*.

## 3.3 Aplicação Web para exibição dos dados

A aplicação web foi desenvolvida em Ruby on Rails e integrada com o servidor de aplicação através de uma API REST. A API foi criada com o serviço *Amazon API Gateway*, em conjunto com o *Amazon Lambda* e *Amazon RDS*. A arquitetura utilizada na solução é apresentada na Figura 3.8.

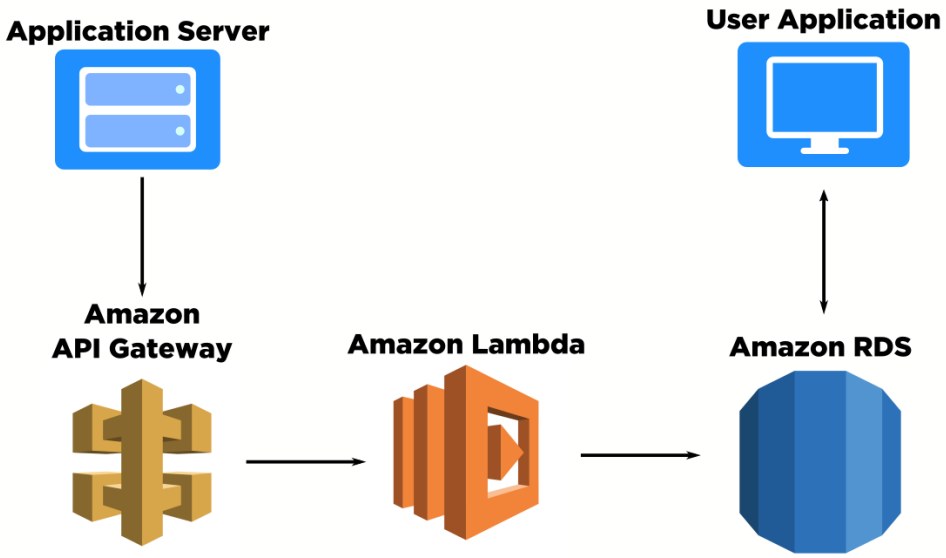


Figura 3.8: Arquitetura da aplicação.

A API consiste de apenas um recurso chamado *node* do tipo POST. As informações acerca do *end-node* são enviadas em JSON e tratadas no *amazon Lambda* de modo que a informação do consumo acumulado seja salva no banco de dados (*Amazon RDS*) estando assim, disponível para a aplicação do usuário. Na Figura 3.9 é apresentado um *screenshot* da aplicação onde o consumo mensal acumulado e o gráfico das seis últimas leituras de consumo energético são gerados com base nos dados fornecidos pelo medidor via LoRaWAN.


Figura 3.9: Aplicação de Gerência de Consumo de Energia (dados ilustrativos).

## 3.4 Resultados

O medidor de energia residencial proposto é capaz de solucionar de forma eficaz os problemas relacionados a leitura de consumo de energia. Os testes com o medidor foram realizados em laboratório durante alguns dias, utilizando como carga uma lâmpada incandescente com potência nominal de 70 Watts. O *end-node*, a cada uma hora, realiza a leitura do consumo acumulado desde a última leitura. A informação extraída do *end-node* é então enviada para o *Amazon Lambda*, através do *Amazon API Gateway*, onde os dados são filtrados e a informação acerca do consumo, salva no banco de dados. Na Figura 3.10, é apresentado um exemplo da estrutura do JSON enviado para o *Amazon API Gateway* com as informações do *end-node*. A informação de consumo é apresentada em base64 no campo *data*. O consumo acumulado da lâmpada a cada uma hora de teste foi de aproximadamente 0.07 kWh, o que condiz com a teoria.

## 3.5 Custo da solução

O *end-node* possui um custo total de US$ 27.88, sendo US$ 18.65 os custos da placa de comunicação, e US$ 9.23 a placa de medição. O *gateway* utilizado, *Sentrius™ RG1xx LoRa-Enabled Gateway*, possui custo em torno de US$ 250. Como o projeto LoRaServer é *opensource*, não existem custos adicionais para sua utilização. Os custos com software e hospedagem podem variar de acordo com a abordagem utilizada pelo desenvolvedor. Na arquitetura apresentada, os custos com software são referentes aos serviços utilizados da *Amazon*:

- *Amazon EC2*: Utilizado para hospedar o Servidor de Rede, Servidor de Aplicação e Aplicação do Usuário. Uma instância *t2.medium* com 4GB de memória e 2 vCPU custando cerca de US$ 0.0464 por hora.
- *Amazon API Gateway*: Para integração com o Servidor de Aplicação com custo de US$ 3,50 por milhão de chamadas de API mais custo de transferência de dados.
- *Amazon Lambda*: Para tratamento dos dados retornados pelo *end-node* com custo de US$ 0,20 por 1 milhão de solicitações, e ultrapassando o primeiro milhão US$ 0,0000002 por solicitação.
- *Amazon RDS*: Para persistência dos dados e posterior disponibilização na aplicação do usuário. Uma instância *db.t2.medium* custa US$ 0,068 por hora.

Considerando que num cenário real os *end-nodes* da rede enviarão os dados de consumo uma vez a cada mês, esta solução se apresenta como uma boa alternativa mesmo quando se trata de milhares de *end-nodes*.

```
{
  "applicationName": "EnergyMeter",
  "nodeName": "DevKit_STM32",
  "txInfo": {
    "frequency": 902900000,
    "adr": true,
    "codeRate": "4/5",
    "dataRate": {
      "modulation": "LORA",
      "bandwidth": 125,
      "spreadFactor": 7
    }
  },
  "fCnt": 15,
  "data": "AAAAAA==",
  "fPort": 4,
  "rxInfo": [{
    "name": "multitech-conduit",
    "loRaSNR": 10,
    "altitude": 87,
    "longitude": -35.77499,
    "mac": "00800000a0000189",
    "latitude": -9.5545,
    "rssi": -45
  }],
  "devEUI": "3431373266365d0d",
  "applicationID": "4"
}
```


Figura 3.10: JSON com informações acerca do *end-node*.

# 4

# Conclusão

A cobertura de uma rede LoRaWAN sobre uma cidade, habilita inúmeras soluções que antes não seriam possíveis ou viáveis com tecnologias populares tais como 3G, 4G, Wifi ou Bluetooth, por exemplo. A área de cobertura, consumo de energia para realizar a comunicação e custos financeiros são fatores que dificultam o desenvolvimento de soluções para cidades inteligentes, e uma rede LoRaWAN viabiliza tais soluções visto que com baixo custo, é possível cobrir cidades inteiras devido a sua capacidade de realizar comunicação a longas distâncias com baixo consumo de energia.

Atualmente, a aferição do uso dos bens de consumo é ineficiente, gerando prejuízos para empresas e uma fraca gestão sobre o serviço prestado para os consumidores. A proposta de solução apresentada, consiste de um *end-node* medidor de energia composto de um módulo LoRaWAN em conjunto com uma placa responsável pela contagem do consumo acumulado. Foi desenvolvida também uma aplicação web a qual tem por objetivo apresentar os dados de consumo acumulado no mês, servindo para demonstrar a viabilidade da solução. Tal proposta, utilizando medidores de energia conectados a uma rede LoRaWAN gera benefícios para ambos os lados, clientes e concessionárias, eliminando os potenciais erros de leitura de modo que a empresa não tenha prejuízos e o usuário pague apenas pelo que foi, de fato, consumido. Este estudo de caso é um exemplo dos inúmeros projetos que podem ser viabilizados através de uma rede LoRaWAN.

## 4.1 Trabalhos futuros

O projeto apresentado se mostra como um esboço de um projeto real, ainda há várias melhorias a serem realizadas, dentre elas: A elaboração de um hardware único, onde o circuito de medição e comunicação estejam juntos, de modo que possam compor um produto viável, completo e compacto; Para comunicação LoRaWAN, é necessário projetar um circuito

de *matching* e trilha para antena, de modo que não haja descasamento de impedância provocando perda de sinal; O software de gestão precisa ser evoluído para um melhor gerenciamento das informações, não apenas para a concessionária mas para o usuário da solução; A criação de uma API REST específica para o consumo das aplicações de usuário, de modo que o banco de dados fique abstraído.

Para que o medidor de energia utilizado atinja uma acurácia de 0.1% conforme apresentado, faz-se necessário uma calibração utilizando alguma referência confiável, como por exemplo, uma fonte de tensão AC 220V real, de modo que não haja oscilações como acontece comumente na rede elétrica. Devido a falta dessa referência, o medidor de energia não foi calibrado adequadamente, sendo assim, as informações de consumo apresentadas neste trabalho não devem ser consideradas com acurácia de 0.1%, mas aproximações do consumo real.

Embora o viés da solução apresentada seja a resolução do problema da leitura do consumo energético, diversos outros ganhos podem ser acrescentados a isso, por exemplo: O usuário poderia acompanhar o próprio consumo diário e verificar alguma anomalia como um consumo excessivo e inesperado; O usuário poderia pelo próprio sistema emitir um boleto com o custo do consumo até determinado dia e não apenas no fim do mês ou ainda escolher a data do vencimento; A concessionária poderia ter maior controle na ativação e desativação do serviço prestado. Além disso, a quantidade de dados gerados por cada *end-node* pode ser explorado de diversas formas, contribuindo para estudos de *big data*, aprendizagem de máquina, entre outros.

# Referências bibliográficas


Semtech sx1301. http://www.semtech.com/wireless-rf/rf-transceivers/sx1301/. Accessed: 2017-10-13.

LoRa Alliance. Lorawan what is it? a technical overview of lora and lorawan. Technical report, November 2015.

Aloÿs Augustin, Jiazi Yi & Thomas Clausen, and William Mark Townsley. A study of lora: Long range and low power networks for the internet of things. *Sensors*, 16(9):1466, September 2016.

Analog Devices. *Single Phase, Multifunction Metering IC with Neutral Current Measurement - Datasheet ADE7953*. Analog Devices, February 2017.

Nicolas Ducrot, Dominique Ray, Ahmed Saadani, Olivier Hersent, Gabor Pop, and Guillaume Remond. Lora device developer guide: Orange connected objects & partnerships, April 2016a. Page: 26. Contributors: Orange and Actility.

Nicolas Ducrot, Dominique Ray, Ahmed Saadani, Olivier Hersent, Gabor Pop, and Guillaume Remond. Lora device developer guide: Orange connected objects & partnerships, April 2016b. Page: 06. Contributors: Orange and Actility.

Nicolas Ducrot, Dominique Ray, Ahmed Saadani, Olivier Hersent, Gabor Pop, and Guillaume Remond. Lora device developer guide: Orange connected objects & partnerships, April 2016c. Page: 7. Contributors: Orange and Actility.

IoT Eclipse.org. The three software stacks required for iot architectures: Iot software requirements and how to implement them using open source technology. Technical report, September 2016.

Ingenu. Ingenu - rpma technology. https://www.ingenu.com/technology/rpma/, a. Accessed: 2017-10-13.

Ingenu. Ingenu - rpma technology. https://www.ingenu.com/technology/rpma/battery-life/, b. Accessed: 2017-11-11.

James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down approach*. Pearson, 6nd edition, 2013.

Link Labs. A comprehensive look at low power, wide area networks. Available at: https://www.link-labs.com/lpwan.

Murata. *LoRa Module Datasheet*. Murata Electronics, February 2017. Revision Code K.

Semtech. *Datasheet: SX1276 - Settings for LoRaWAN V2.0*. Semtech Corporation.

Semtech. *AN1200.28 LoRa MCU Specifications and Requirements*. Semtech Corporation, Jul 2015a. revision 2.

Semtech. *AN1200.22 LoRa Modulation Basics*. Semtech Corporation, May 2015b. revision 2.

Weightless SIG. Weightless. www.weightless.org. Accessed: 2017-10-13.

Sigfox. Sigfox technology overview. https://www.sigfox.com/en/sigfox-iot-technology-overview. Accessed: 2017-10-13.

N. Sornin, M. Luis, T. Eirich, T. Kramp, and O.Hersent. *LoRaWAN Specification*. LoRa Alliance, July 2016. Version 1.0.2; Status: final.